Investigating the World of Pokémon

Author: Matthew W. Noble [About the Author]

Date: 2017 · December · 02

Project Overview

The aim of the following project is to demonstrate my coding and report writing capabilities within a fun topic that I know inside and out: Pokémon. My dream job is to be a Data Scientist; although I have a very strong educational background[1] and a proven track record of self-motivated research, I don't have many examples of my work (besides my DPhil project and the resulting thesis) to demonstrate my skill set and sell myself with. This project is intended to be the first of many in what will grow to be my online data science portfolio.

If you would like to contact me and give me a job because you loved everything about this project, or you are a more experienced data scientist and would be willing to offer advice on how I can become better, please see the About the Author section at the end of this report.

Let's begin … are you a boy or are you a girl?

Pokémon: A Brief History

Pokémon is a media franchise managed by The Pokémon Company, a Japanese consortium between Nintendo, Game Freak, and Creatures. The franchise was created by Satoshi Tajiri in 1995, and is centred on fictional creatures called "Pokémon", which humans, known as "Pokémon Trainers", catch and train to battle each other for sport.

[1] **BSc** (Hons.) in **Physics** from the Department of Physics, **University of St Andrews**.
MSc (Merit) in **Fusion Energy** from the York Plasma Institute, **University of York**.
DPhil in **Materials Science** from the Department of Materials Science, **University of Oxford**.

[Source: https://en.wikipedia.org/wiki/Pok%C3%A9mon]

Video Games

Generations

The original Pokémon games were role-playing games (RPGs) with an element of strategy and were created for the Game Boy. These RPGs, and their sequels, remakes, and English language translations, are still considered the "main" Pokémon games, and the games which most fans of the series are referring to when they use the term "Pokémon games". All of the licensed Pokémon properties overseen by The Pokémon Company International are divided roughly by generation. These generations are roughly chronological divisions by release; every several years, when an official sequel in the main RPG series is released that features new Pokémon, characters, and gameplay concepts, that sequel is considered the start of a new generation of the franchise. The main games and their spin-offs, the anime, manga, and trading card game are all updated with the new Pokémon properties each time a new generation begins. The franchise began the seventh generation on November 18, 2016.

[Source: https://en.wikipedia.org/wiki/Pok%C3%A9mon]

These generations, to over simplify, are defined by the following video games:

Generation 1:	Red, Blue, and Yellow
Generation 2:	Gold, Silver, and Crystal
Generation 3:	Rugby, Sapphire, Emerald, FireRed, and LeafGreen
Generation 4:	Diamond, Pearl, Platinum, HeartGold, and SoulSilver
Generation 5:	Black, White, Black 2, and White 2
Generation 6:	X, Y, Omega Ruby, and Alpha Sapphire
Generation 7:	Sun, Moon, Ultra Sun, and Ultra Moon

[Source: https://bulbapedia.bulbagarden.net/wiki/Generation]

Game mechanics

The main staple of the Pokémon video game series revolves around the catching and battling of Pokémon. Starting with a starter Pokémon, the

player can catch wild Pokémon by weakening them and catching them with Poké Balls. Conversely, they can choose to defeat them in battle in order to gain experience for their Pokémon, raising their levels and teaching them new moves. Most Pokémon have "evolution families", a term which refers to the Pokémon to evolve into or be evolved into more powerful forms by raising their levels or using certain items. Throughout the game, players will have to battle other trainers in order to progress, with the main goal to defeat various Gym Leaders/Trials and earn the right to become the regional champion. Subsequent games in the series have introduced various side games and side quests, including the Battle Frontiers that display unique battle types and the Pokémon Contests where visual appearance is put on display.

[Source: https://en.wikipedia.org/wiki/Pok%C3%A9mon]

Step 0: Collecting the Data

Attempt 0: Web-Scrapping

Initially, I tried to collect my own data via web-scraping using the Requests, urllib, and BeautifulSoup libraries, however, I encountered several issues, the main one being that the websites with said data, such as:

https://www.serebii.net/pokedex-sm/001.shtml
https://bulbapedia.bulbagarden.net/wiki/Bulbasaur_(Pokémon)
https://www.pokemon.com/us/pokedex/bulbasaur
https://pokemondb.net/pokedex/bulbasaur

did not possess a constant layout of their information. In the sense that the "base attack" stat may be in cell[1099] of the website's HTML for Bulbasaur but it would be in cell[902] for Ivysaur. A lot of these issues were the result of different Pokémon having different attacks and different abilities and ... etc. which is perfectly acceptable to layout a website to fit your data, but I struggled to then find fixed pointers in the HTML that I could use to navigate and hence I found I was writing in more exceptions to my code than normal runs. It was legitimately faster to open the respective Pokémon's web page and copy/paste the information by hand.

This task highlighted an area of future development, I'll move forward by reading more on the subject and most likely dedicate an entire project to exclusively data collection, but for the sake of this project:

Attempt 1: Kaggle

Browsing through the data sets on Kaggle, there are several dedicated to Pokémon. For no particular reason, I chose the following one:

https://www.kaggle.com/alopez247/pokemon

and downloaded the following data file:

pokemon_alopez247.csv

The data file was renamed to:

PokemonData.csv

and saved in my working directory for the Pokémon project. According to the Kaggle information blurb, the fields include:

- **Number:**
  Pokémon ID in the National Pokédex.
- **Name:**
  Name of the Pokémon.
- **Type_1:**
  Primary type.
- **Type_2:**
  Secondary type, in case the Pokémon has it.
- **Total:**
  Sum of all the base stats (Health Points, Attack, Defence, Special Attack, Special Defence, and Speed).
- **HP**:
  Base Health Points.
- **Attack:**
  Base Attack.

- **Defense:**
  Base Defence.
- **Sp_Atk:**
  Base Special Attack.
- **Sp_Def:**
  Base Special Defence.
- **Speed:**
  Base Speed.
- **Generation:**
  Number of the generation when the Pokémon was introduced.
- **isLegendary:**
  Boolean that indicates whether the Pokémon is Legendary or not.
- **Color:**
  Colour of the Pokémon according to the Pokédex.
- **hasGender:**
  Boolean that indicates if the Pokémon can be classified as male or female.
- **Pr_male:**
  In case the Pokémon has Gender, the probability of its being male. The probability of being female is, of course, 1 minus this value.
- **Egg_Group_1:**
  Egg Group of the Pokémon.
- **Egg_Group_2:**
  Second Egg Group of the Pokémon, in case it has two.
- **hasMegaEvolution:**
  Boolean that indicates whether the Pokémon is able to Mega-evolve or not.
- **Height_m:**
  Height of the Pokémon, in meters.
- **Weight_kg:**
  Weight of the Pokémon, in kilograms.
- **Catch_Rate:**
  Catch Rate.
- **Body_Style:**
  Body Style of the Pokémon according to the Pokédex.

Opening the PokemonData.csv file, the data is a mixture of integers, Booleans, and strings, but each column is consistent. The header is present

and all of the header titles are as described in the data information blurb. The data's delimiter is a "," being a .csv.

The data was imported into a Pandas data frame. The NationalDex numbers (Number) of the Pokémon were used to index the entries.

Step 1: Data Analysis

Inspection[2]

The first step of any investigation should be inspection. After loading the .csv file and creating a Pandas data frame, I confirmed that the field headings were correct and that the information entered was easily readable and consistent. The data set was small enough that it did not need to be imported and analysed in chunks.

The next check was for `NaN` entries. I used my `NaN_Checker` and `NaN_Counter` functions within my `HomeBrewFunctions` suite to perform this task. The `NaN_Checker` informed me that `NaN` values existed within the pandas data frame and the `NaN_Counter` told me that the columns housing them were Type_2, Pr_Male, and Egg_Group_2 with 371, 77, and 530 entries respectively. `NaN` values in these fields are fine. Regarding a Pokémon's type, a Pokémon can be either a pure typing e.g. pure Electric like Pikachu or it can be dual typed e.g. Grass and Poison like Bulbasaur. Regarding a Pokémon's probability of being male, there are two factors at play here, 1) some Pokémon can only be male e.g. Tauros or can only be female e.g. Jynx, and 2) some Pokémon do not have a gender, e.g. Porygon. Regarding the egg groups, sometimes a Pokémon only belongs to one egg group and hence the second is `NaN`.

[2] The code for this section can be found here: `InitialInspection.py`

Exploration and Visualisation[3]

Computing the length of the data frame reveals our first insight, there are a total of 721 Pokémon (as of Generation 6). These 721 Pokémon can be sub-divided by Generation, as shown in figure 01.

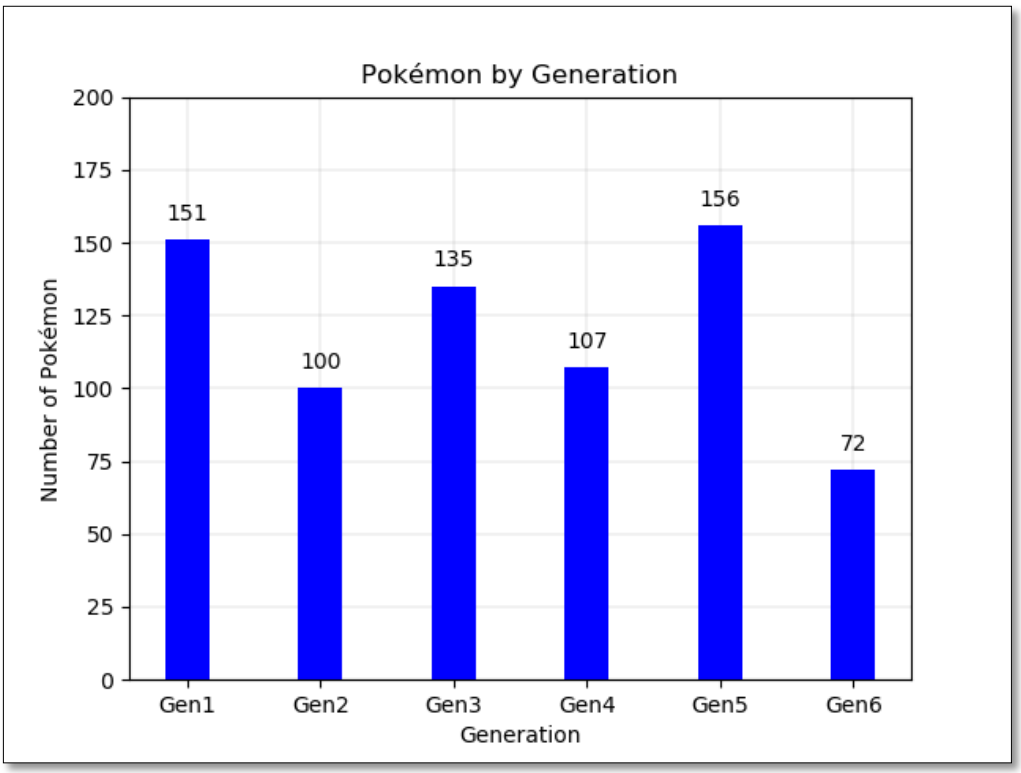


Figure 01: Bar Chart showing the total number of Pokémon in each Generation. Plot created using `GenerationPlot.py`

The initial exploration began by invoking `.describe()` and `.corr()` on the data frame and passing the columns of the data frame through the `UniqueWordCounter` function I originally wrote to analyse tweet data. The `UniqueWordCounter` function is a part of my `HomeBrewFunctions.py` collection.

Types

There are a total of 18 Pokémon types, they are:

[3] Unless noted further, the code for this section can be found here: `ExplorationAndVisualisation.py`

- Grass
- Fire
- Water
- Bug
- Normal
- Poison
- Electric
- Ground
- Fairy
- Fighting
- Psychic
- Rock
- Ghost
- Ice
- Dragon
- Dark
- Steel
- Flying

The primary, secondary, and combined primary and secondary type distributions of all the Pokémon over all the generations are shown in figure 02.

The same type distribution plots were also created for the individual generations. For the sake of space and document flow, they can be found in appendix A.

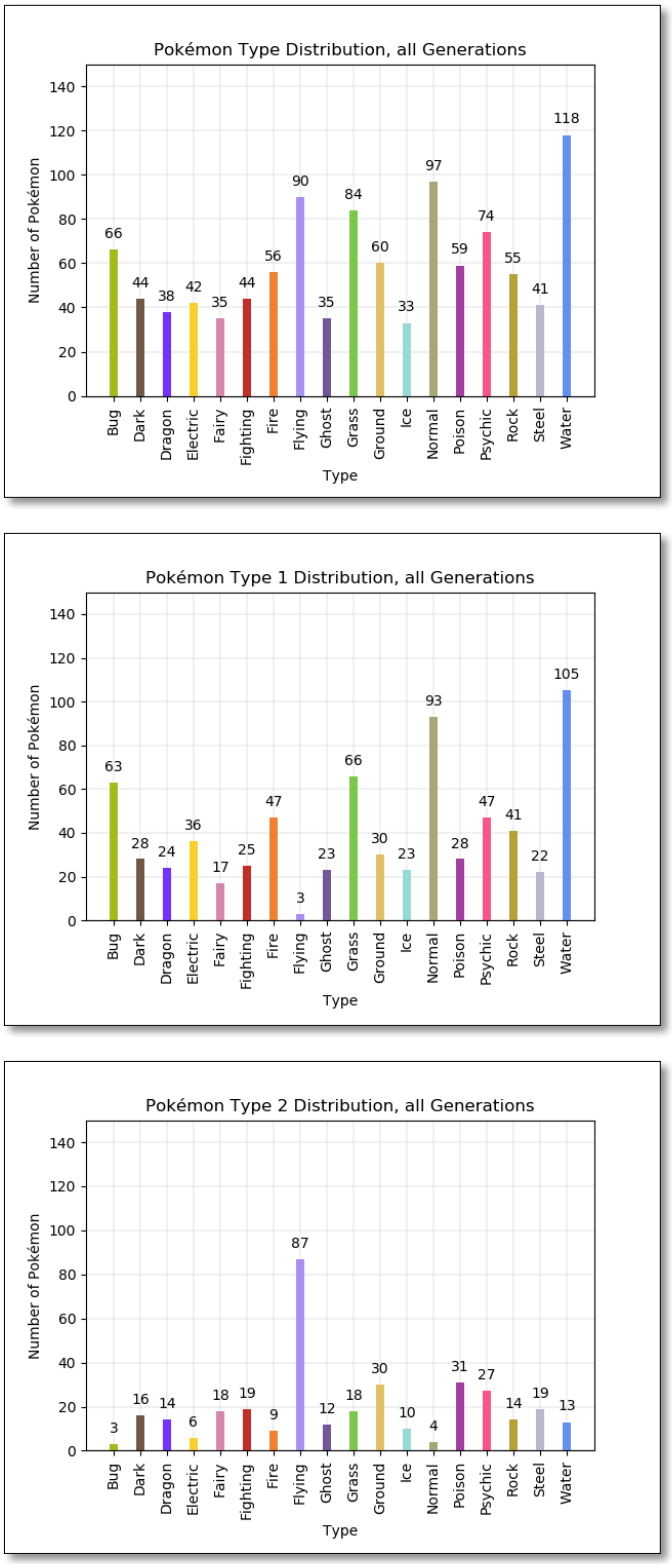


Figure 02: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for all generations. Plots created using `TypePlot.py`

Some interesting insights:

- Overall, the water type is the most common typing whereas the ice type is the least common.
- The water type is the most common primary typing whereas the flying type is the least common.
- The flying type is the most common secondary typing whereas the bug type is the least common.
- With 3 exceptions, the flying type is only a secondary typing.
- The fairy, flying, ground (tied), and poison types all have more secondary typing's than primary typing's.

Looking at the individual generation spreads the one thing that struck me as interesting was generation 1 doesn't have any dark types.

Base Stats

The base stats of the Pokémon are a measure of how strong they are. The higher the number, the more HP a Pokémon has or the faster they are or the more resilient to damage they are etc. The stats are separated into 6 categories:

- HP (Hit Points)
- Attack
- Defence
- Special Attack
- Special Defence
- Speed

The attack and defence are applicable to physical moves e.g. bullet punch or horn attack. The special attack and special defence stats are applicable to non-physical moves, such as psychic or flamethrower. A general rule of thumb states: if a Pokémon needs to make contact, then the move is physical, if not, then it is special. The total is the sum of all the stats and acts as a general indicator of how strong the Pokémon is overall.

A Pearson r correlation was done on the stats, the results are shown in table 01.

	Tot.	HP.	Att.	Def.	Sp.A.	Sp.D.	Spe.
Tot.	1.000	0.643	0.704	0.606	0.724	0.707	0.549
HP	0.643	1.000	0.432	0.229	0.369	0.376	0.170
Att.	0.704	0.432	1.000	0.433	0.335	0.207	0.335
Def.	0.606	0.229	0.433	1.000	0.203	0.484	−0.009
Sp.A.	0.724	0.369	0.335	0.203	1.000	0.493	0.443
Sp.D.	0.707	0.376	0.207	0.484	0.493	1.000	0.233
Spe.	0.549	0.170	0.335	−0.009	0.443	0.233	1.000

Table 01: Pearson r correlation matrix generated from the base stats. Small, medium, and large associations are colour coded.

A good rule for the Pearson r correlation states that a correlation between 0.10 and 0.29 represents a small association or relationship, coefficients between 0.30 and 0.49 represent a medium association or relationship, and coefficients of 0.50 and above represent a large association or relationship. For the values in table 01, the results have been shaded accordingly to reflect small, medium, and large associations.

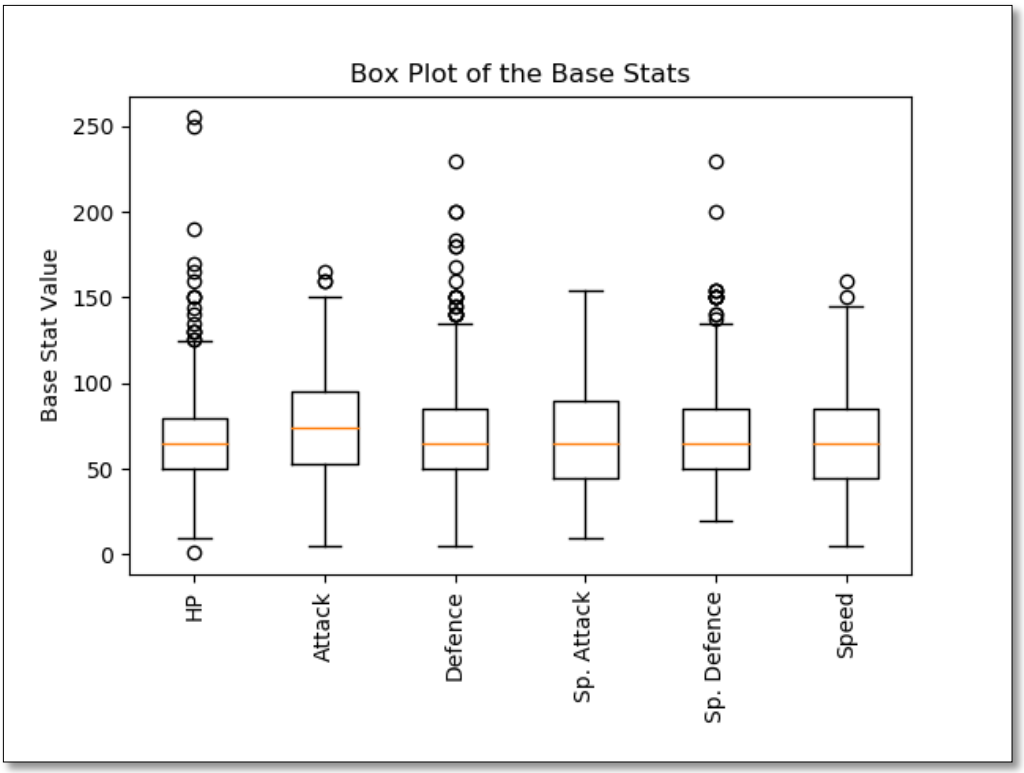
Of all the results, total being highly associated with all of the other base stats makes sense since it is the sum of all the stats. It is most associated with special attack and least with speed. Speed and defence are the only entries that have no - not even a small - association, in fact, they are negatively associated.

The bases stats were plotted using a box and whisker plot. This is shown in figure 03.

Using `.describe()` on the base stats generates table 02.

	Tot.	HP.	Att.	Def.	Sp.A.	Sp.D.	Spe.
mean	418	68	75	71	69	69	66
std	110	26	29	29	29	27	27
min	180	1	5	5	10	20	5
25%	320	50	53	50	45	50	45
50%	424	65	74	65	65	65	65
75%	499	80	95	85	90	85	85
max	720	255	165	230	154	230	160

Table 02: Description of the base stats including the mean, std, max, min, and quarter-quartiles.



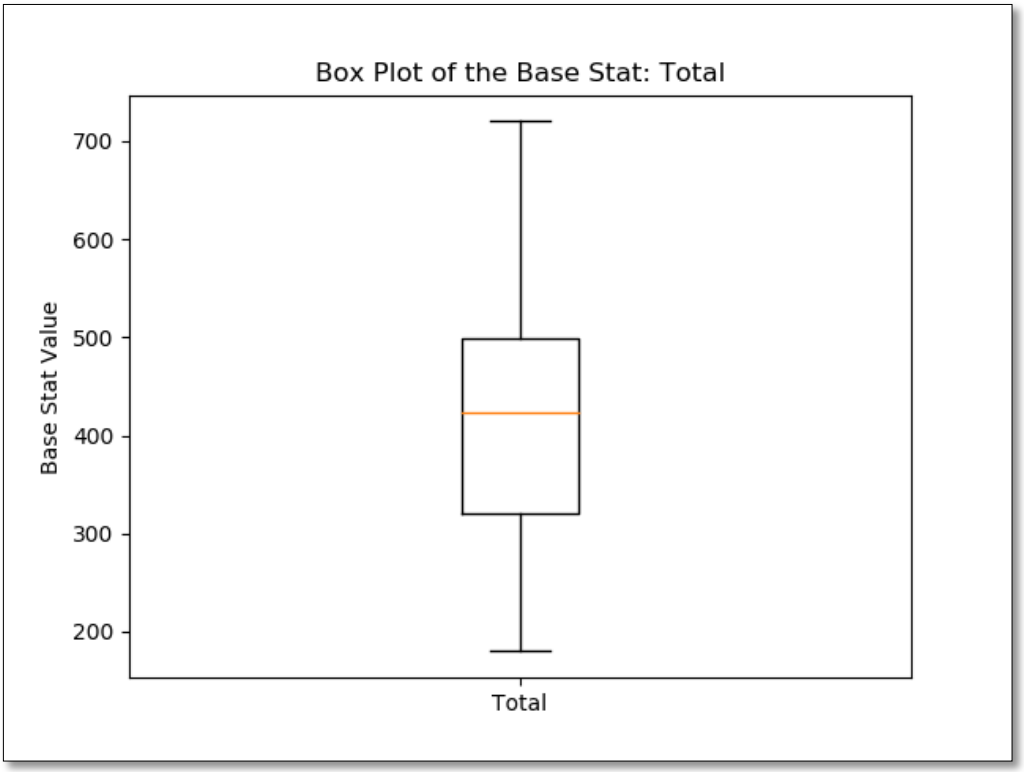


Figure 03: Box plots of the Pokémon base stats and the Pokémon base stat totals. Plot created using `StatsPlot.py`

The top 5 Pokémon (in descending order) for each stat (noted in [*]) are:

Total:
Arceus [720], Mewtwo [680], Lugia [680], Ho-Oh [680], and Rayquaza [680]

HP:
Blissey [255], Chansey [250], Wobbuffet [190], Wailord [170], and Alomomola [165]

Attack:
Rampardos [165], Slaking [160], Regigigas [160], Groudon [150], and Rayquaza [150]

Defence:
Shuckle [230], Steelix [200], Regirock [200], Avalugg [184], and Cloyster [180]

Sp. Attack:
Mewtwo [154], Kyogre [150], Rayquaza [150], Deoxys [150], and Dialga [150]

Sp. Defence:
Shuckle [230], Regice [200], Lugia [154], Ho-Oh [154], and Florges [154]

Speed:
Ninjask [160], Deoxys [150], Accelgor [145], Electrode [140], and Jolteon [130]

The bottom 5 Pokémon (in ascending order) for each stat (noted in [*]) are:

Total:
Sunkern [180], Azurill [190], Kricketot [194], Caterpie [195], and Weedle [195]

HP:
Shedinja [1], Diglett [10], Magikarp [20], Pichu [20], Shuckle [20]

Attack:
Chansey [5], Happiny [5], Magikarp [10], Shuckle [10], and Blissey [10]

Defence:

Chansey [5], Happiny [5], Blissey [10], Abra [15], and Pichu [15]

Sp. Attack:
Shuckle [10], Feebas [10], Bonsly [10], Magikarp [15], and Happiny [15]

Sp. Defence:
Caterpie [20], Weedle [20], Magikarp [20], Igglybuff [20], and Carvanha [20]

Speed:
Shuckle [5], Munchlax [5], Trapinch [10], Bonsly [10], and Ferroseed [10]

Legendary

There are a total of 48 legendary Pokémon across the 6 generations. The legendries tend to be the Pokémon on the box art or the Pokémon the movies are based around. Their distribution across the generations is shown in figure 04.

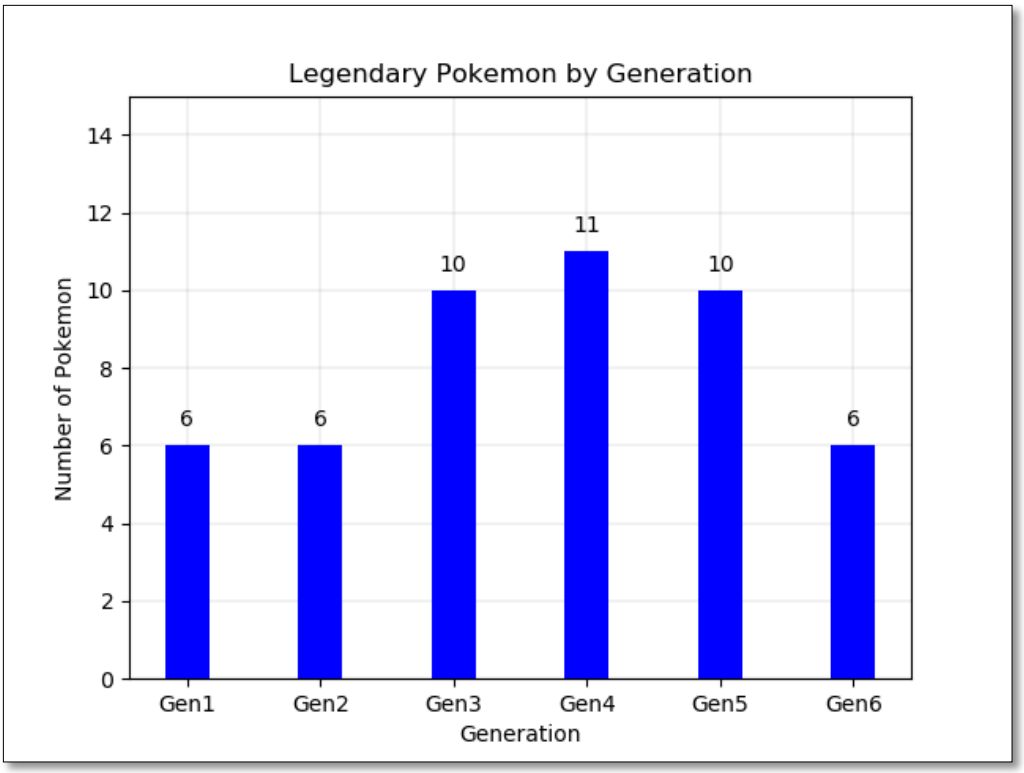


Figure 04: Bar chart showing the number of legendary Pokémon per generation. Plot created using `LegendaryPlot.py`

Querying the reduced data frames to return their names, the legendary Pokémon by generation are:

Generation 1
Articuno, Zapdos, Moltres, Mewtwo, and Mew

Generation 2
Raikou, Entei, Suicune, Lugia, Ho-Oh, and Celebi

Generation 3
Regirock, Regice, Registeel, Latias, Latios, Kyogre, Groudon, Rayquazza, Jirachi, and Deoxys

Generation 4
Uxie, Mespirit, Azelf, Dialga, Palkia, Heatran, Regigigas, Giratina, Darkrai, Shaymin, and Arceus

Generation 5
Victini, Cobalion, Terrakion, Virizion, Tornadus, Thundurus, Reshiram, Zekrom, Landorus, and Kyuren

Generation 6
Xerneas, Yveltal, Zygarde, Diancie, Hoopa, and Volcanion

Looking at their data entries, the legendary Pokémon tend to have three defining characteristics:

1. They tend to be un-breedable (Egg_Group_1 == Undiscovered)
2. They tend to be genderless (hasGender == False)
3. They tend to be powerful (Total > 600)

The exceptions to the sup 600 total rule are:

Articuno, Zapdos, Moltres, Raikou, Entei, Suicune, Regirock, Regice, Registeel, Uxie, Mespirit, Azelf, Cobalion, Terrakion, Virizion, Tornadus, and Thundurus.

All of the sub 600 total Pokémon have the same total of 580, which seems a strange coincidence. It also appears that they (with the exception of the missing Landorus) are the legendary trios of their respective generations.

[Source: http://pokemon.wikia.com/wiki/Legendary_Trio]

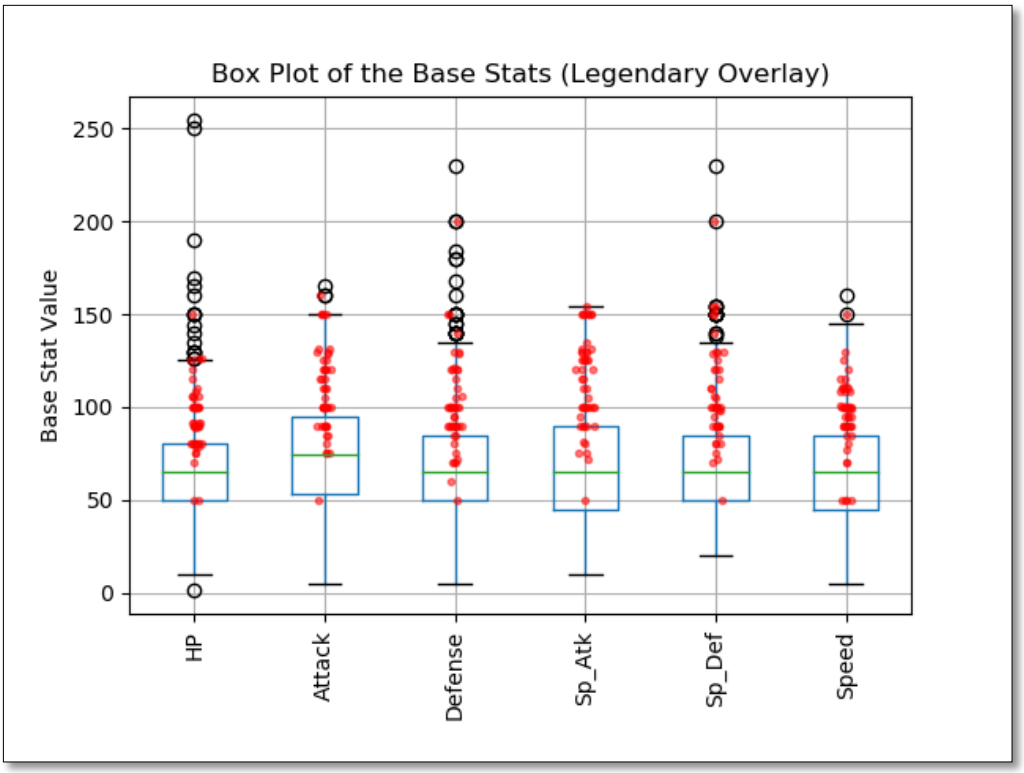
The exceptions to the agendered rule are:

Latias, Latios, Heatran, Tornadus, Thundurus, and Landorus.

Latias is always a girl, Latios, Tornadus, Thundurus, and Landorus are always boys, and Heatran has a 50:50 chance of being either.

There are no exceptions to the Egg_Group rule.

Figure 05 shows the legendary stats overlaid on top of a box plot of all of the Pokémon base stats and the Pokémon base stat totals. The legendary Pokémon are clearly above average.



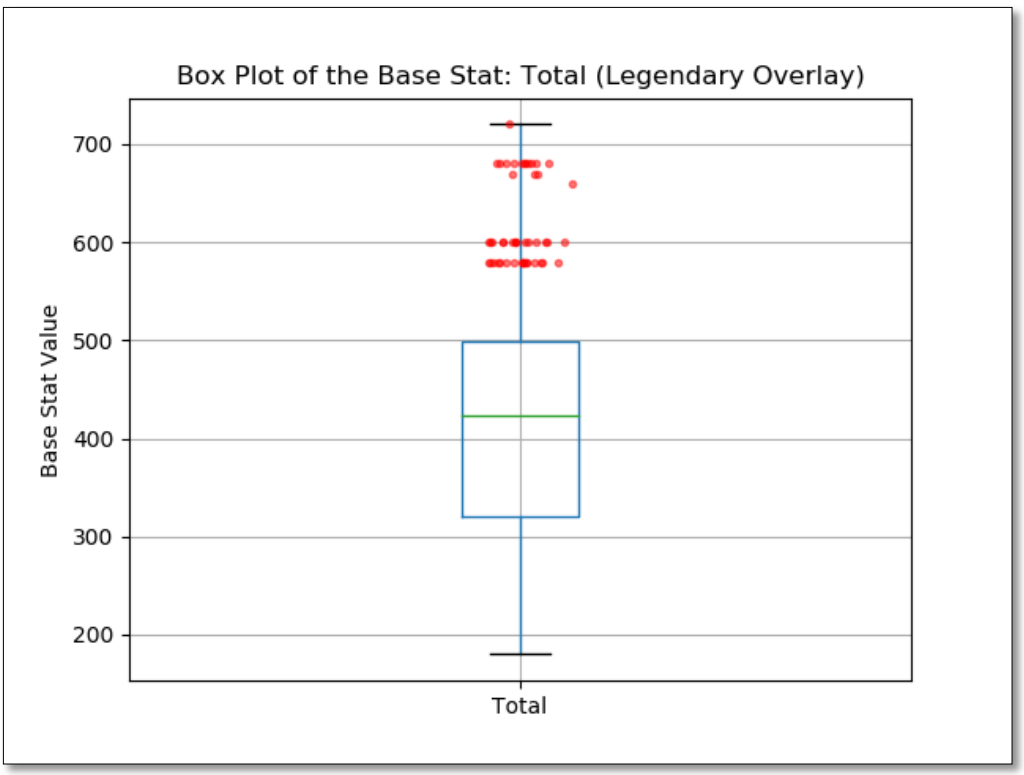


Figure 05: Box plots of all Pokémon stats with the legendary Pokémon's stats over laid in red. The jitter in the x-direction was to avoid over-crowding the centre and is purely aesthetic. Plots created using `LegendaryPlot.py`

Colour

The colour of a Pokémon is a general grouping that denotes the majority of the colour palette within a Pokémon's sprite. There are a total of 10 colour groups, they are listed in figure 06 along-side their frequency.

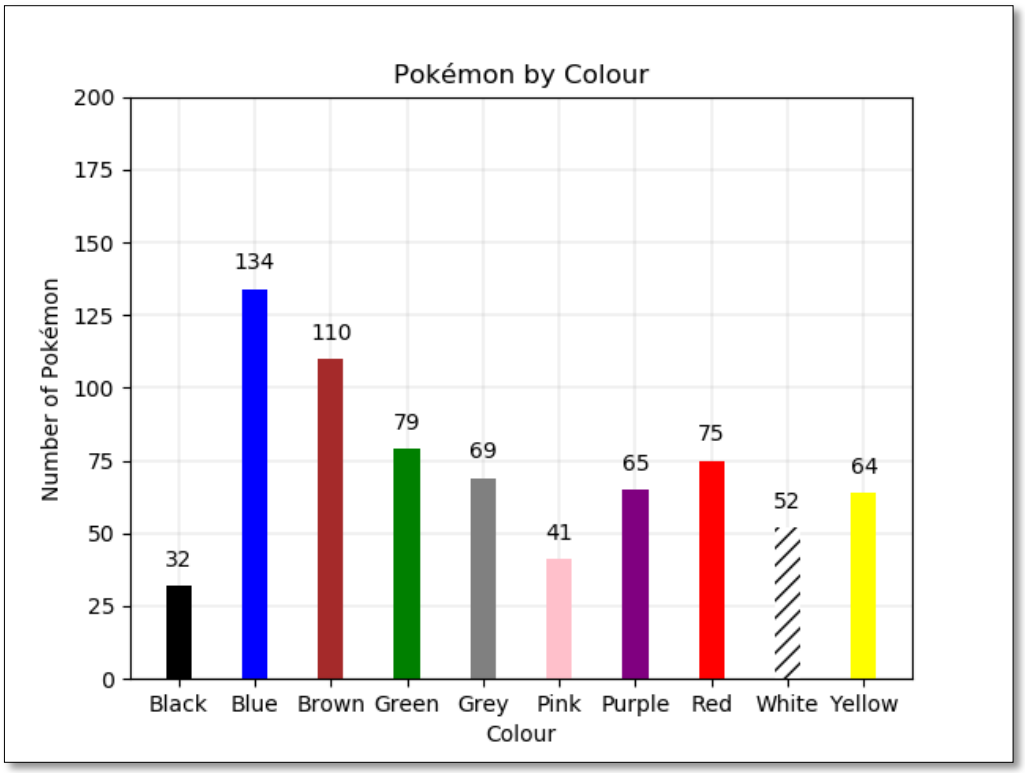
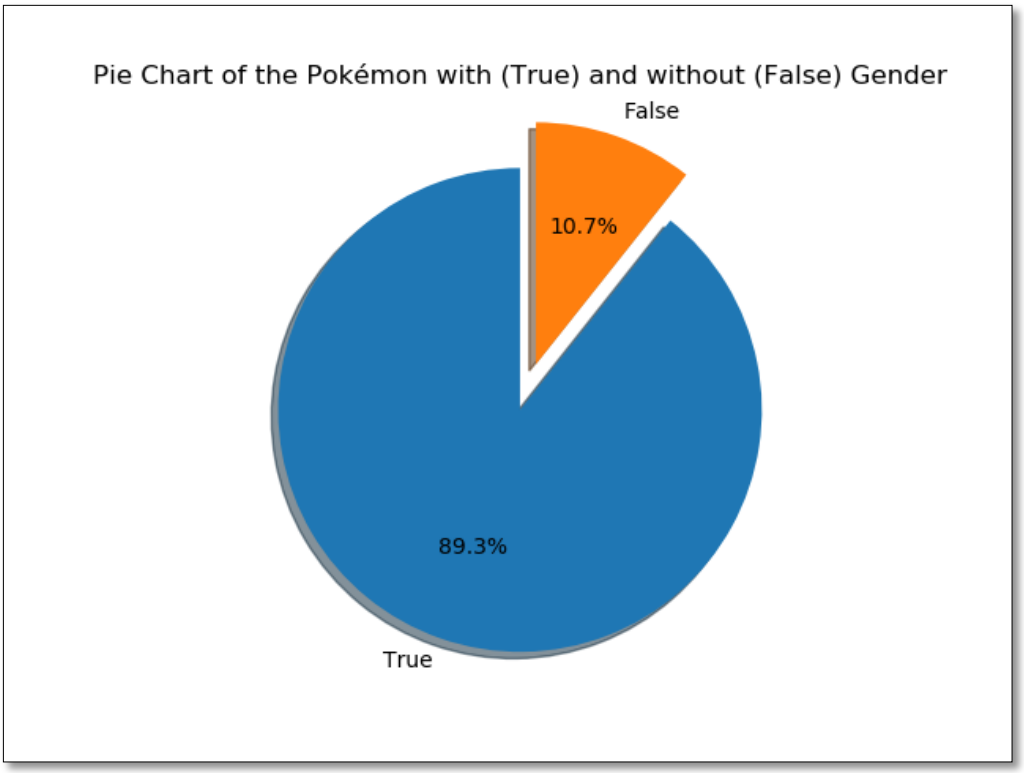


Figure 06: Bar chart of all Pokémon colours. Plots created using `ColourPlot.py`

Gender and Probability of Being Male

Of the 721 Pokémon, 644 are classified as having a gender compared to 77 which are classified as not. The probability of being a male is given in the "Pr_Male" column. The probability of being a female is therefore $1 - P(Male)$. The $P(Male)$ only takes 8 values with a 50:50 chance of being either male or female being the most common state. This is shown in figure 07.



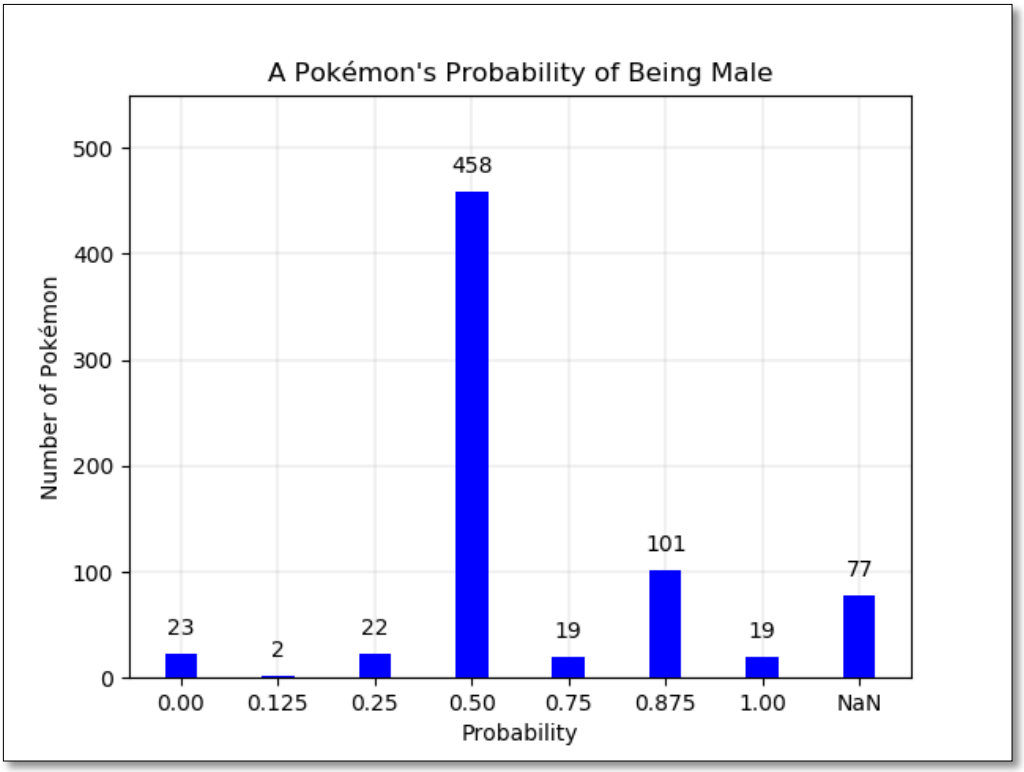


Figure 07: Pie chart and a bar chart showing 1) the amount of Pokémon with a gender Vs those without and 2) the distributions of the probability of Pokémon being male. Plots created using `GenderPlot.py`

Egg Group

There are a total of 15 egg groups if one counts “Undiscovered” and “Ditto”. The egg groups and their frequency are shown in figure 08. The egg group characteristic of a Pokémon is linked to the in-game breeding mechanic. Pokémon can only breed in the day-care and produce an egg if:

- They are not legendary Pokémon, baby Pokémon, Unown, Nidorina or Nidoqueen.
- They are of opposite genders.
- They are in the same egg group (see link).

A Pokémon meeting the first criteria can also breed with Ditto. Genderless Pokémon can only breed with Ditto.

More information including Exceptions, Passing down moves, Passing down IV stats, Baby Pokémon, and Egg group connections can be found here:

[Source: https://pokemondb.net/mechanics/breeding]

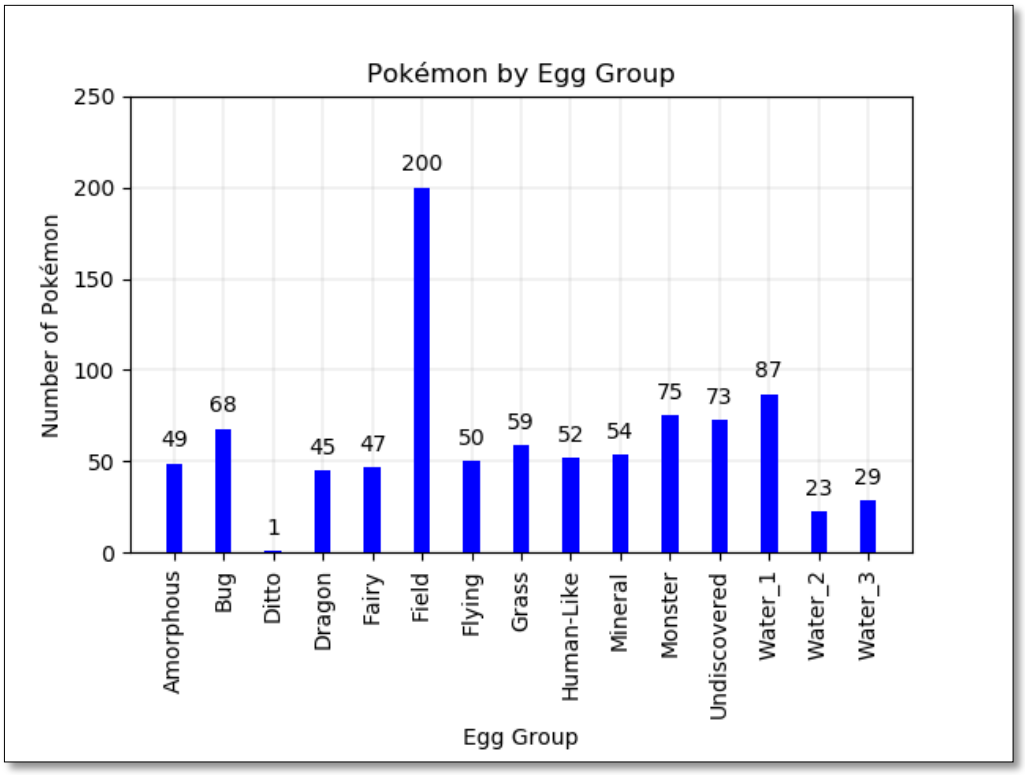


Figure 08: Bar chart of all Pokémon egg groups. Plots created using `EggGroupPlot.py`

Body Style

A Pokémon's body style is linked to their Pokédex entry. It is based on their silhouette and the games use different categories or markers depending on the generation. The names themselves are therefore fan-created to transfer the shapes into a more rigorously defined set of categories. These names and their frequency are shown in figure 09.

[Source: https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_body_style]

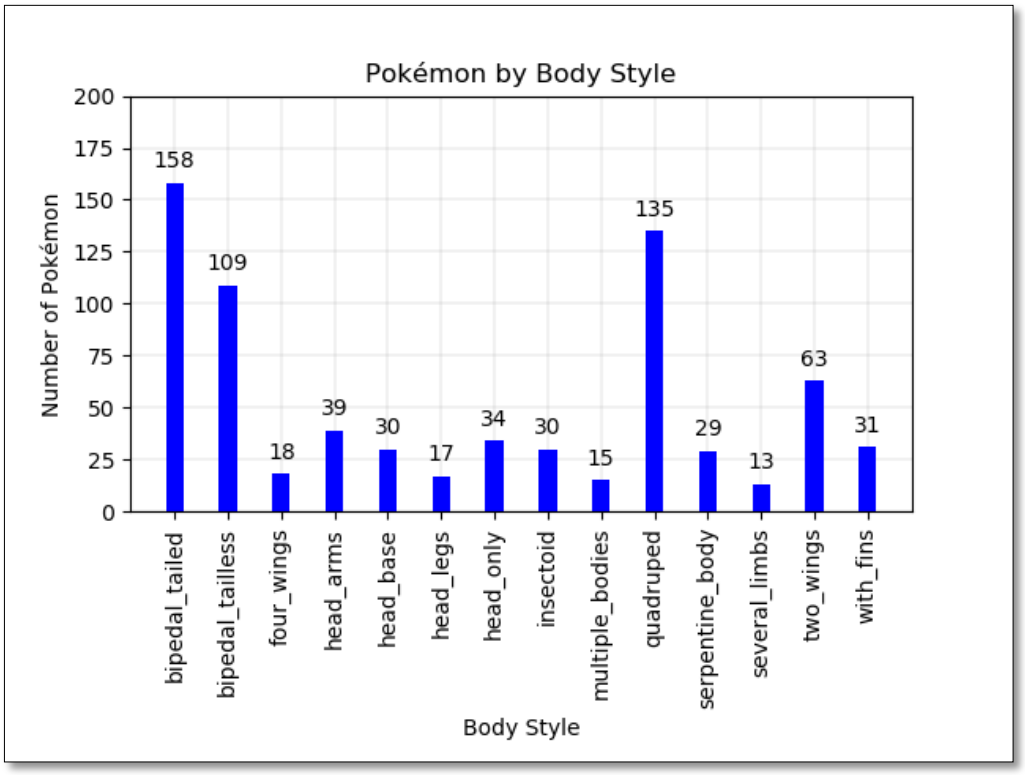
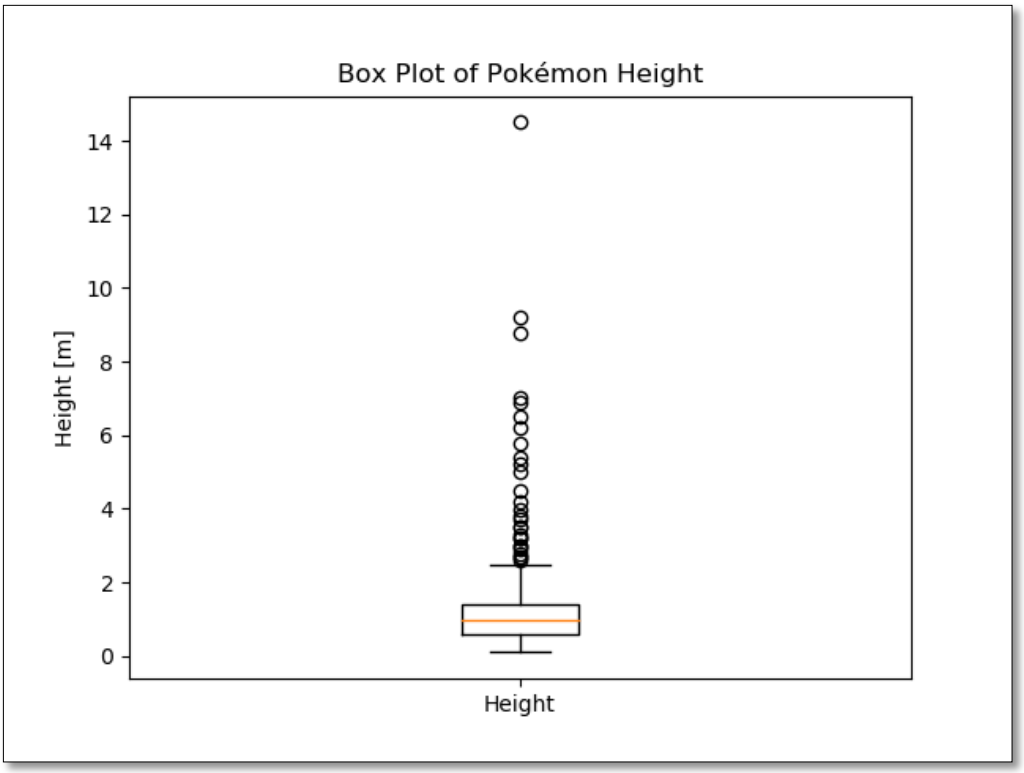


Figure 09: Bar chart of all Pokémon body styles. Plots created using `BodyStyle.py`

Height and Weight

The height and weight of the Pokémon were plotted using box plots in figure 10 and a scatter plot in figure 11.



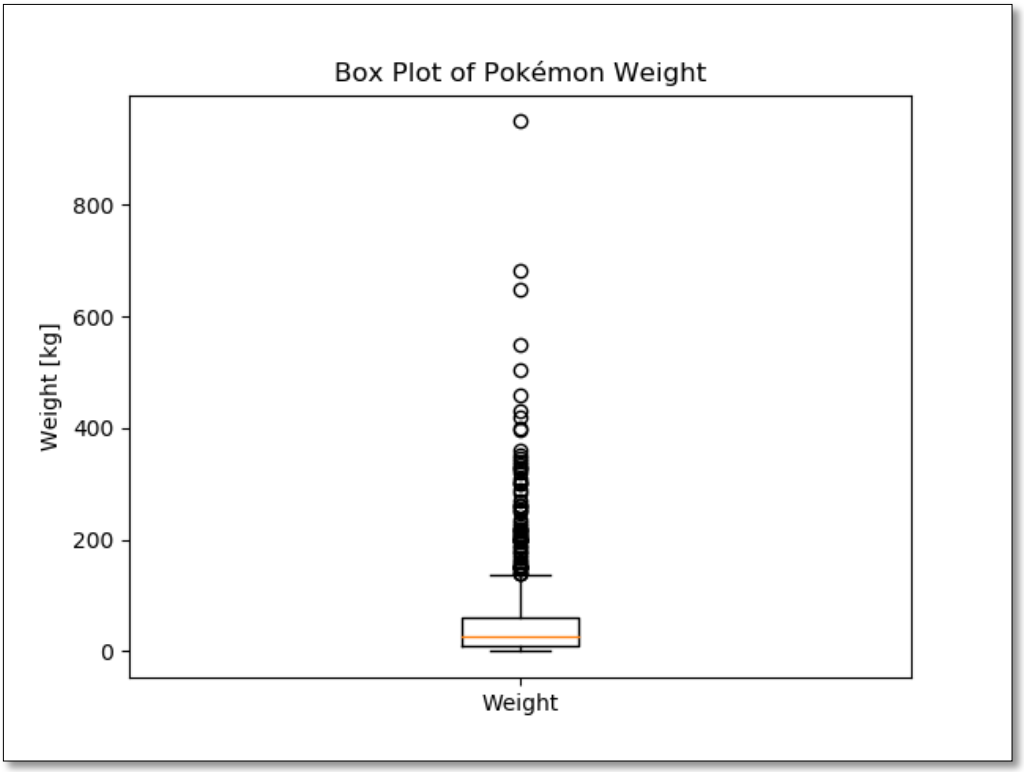


Figure 10: Box plots of all Pokémon's height and weight. Plots created using `HeightAndWeight.py`

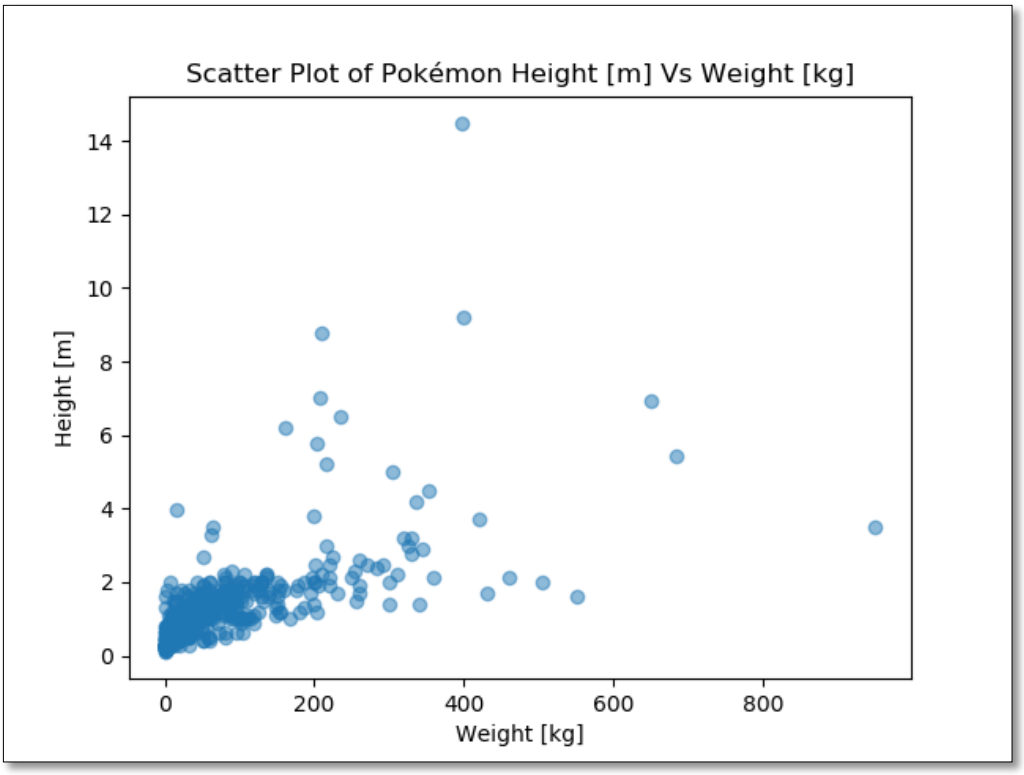


Figure 11: Scatter plot of Pokémon heights Vs Pokémon weights. Plots created using `HeightAndWeight.py`

The `.describe()` function was used on the height and weight columns, it generated the following table 03.

	Height [m]	Weight [kg]
mean	1.14	56.77
std	1.04	89.10
min	0.10	0.10
25%	0.61	9.40
50%	0.99	28.00
75%	1.40	61.00
max	14.50	950.00

Table 03: Description of the height and weight including the mean, std, max, min, and quarter-quartiles.

The heights and weights appear to be positively correlated. A Pearson r correlation matrix was created to investigate this. The Pokémon stats were also included to see if there was any correlation with a Pokémon's physical size and their power. The results are shown in table 04.

	Tot.	HP.	Att.	Def.	Sp.A.	Sp.D.	Spe.	Hei.	Wei.
Tot.	1.000	0.643	0.704	0.606	0.724	0.707	0.549	0.527	0.536
HP	0.643	1.000	0.432	0.229	0.369	0.376	0.170	0.443	0.431
Att.	0.704	0.432	1.000	0.433	0.335	0.207	0.335	0.409	0.469
Def.	0.606	0.229	0.433	1.000	0.203	0.484	−0.009	0.354	0.477
Sp.A.	0.724	0.369	0.335	0.203	1.000	0.493	0.443	0.331	0.285
Sp.D.	0.707	0.376	0.207	0.484	0.493	1.000	0.233	0.313	0.329
Spe.	0.549	0.170	0.335	−0.009	0.443	0.233	1.000	0.225	0.109
Hei.	0.527	0.443	0.409	0.354	0.331	0.313	0.225	1.000	0.661
Wei.	0.536	0.431	0.469	0.477	0.285	0.329	0.109	0.661	1.000

Table 04: Pearson r correlation matrix generated from the Pokémon's base stats and their height and weight. Small, medium, and large associations are colour coded.

A Pokémon's height and weight are strongly associated with one another, they are also strongly associated with the stat total, indicating that the larger the Pokémon is, the more powerful it is. There is a medium association for all of the stats with height and weight with the exceptions of special attack and speed. The weaker association with speed implies that the larger Pokémon are slower, this seems obvious, the more curious one is weight and special attack, which implies the lighter Pokémon favour the non-physical special moves, perhaps to compensate for their reduced physicality?

Step 2: Machine Learning

The next task, now that the data has been thoroughly explored, is to see what questions can be asked of it. Usually, one would have their question in mind and collect the data accordingly, but since this data was provided, we shall make-do.

Due to the nature of the Pokémon games and how their stats are coded in, asking questions such as: Can you predict the colour? Or: Can you predict the gender? Are not going to be particularly useful without (I imagine) hard-coding in various hyper-parameters to account for the seemingly random assignment of those characteristics. A far simpler question, and I believe an easily answerable one is: Is it legendary?

The model will accept 2 fields of data:[4] a Pokémon's total and their Gender. These two fields, as shown above, tend to imply a legendary Pokémon. The model will then accept a label: Legendary or Non-Legendary (True or False).

The SciKit-Learn package was imported, and from that, the `tree`, `Perceptron`, `KNeighborsClassifier`, and `GaussianNB` models. The `accuracy_score` was also imported to test the predictions.

The code is available in `MachineLearningLegendary.py` and returned the following accuracy scores:

DecisionTree_Accuracy:	98.75173370319001
Perceptron_Accuracy:	93.34257975034674
KNN_Accuracy:	98.47434119278779
Gauss_Accuracy:	96.67128987517337

The best performer was the DecisionTree. None of the model's hyper-parameters were changed to make these predictions.

Finally, what about Generation 7 and Pokémon the models don't know the answers too? Rather than creating a data-set for Generation 7, I went onto Serebii and searched for a hand-full of Pokémon which would test the model fairly. Four Pokémon were selected:

Solgaleo is one of the box-legendries. It has a total of 680 and is genderless. [https://bulbapedia.bulbagarden.net/wiki/Solgaleo_(Pok%C3%A9mon)]

Kommo-o is a very strong non-legendary Pokémon. It has a total of 600 and has a gender. [https://bulbapedia.bulbagarden.net/wiki/Kommo-o_(Pok%C3%A9mon)]

Tapu Koko is one of the legendary trio of that generation. It has a total of 570 and is genderless. [https://bulbapedia.bulbagarden.net/wiki/Tapu_Koko_(Pok%C3%A9mon)]

[4] It was originally going to accept a third: Egg_Group_* but it didn't like the strings. This can be rectified by assigning the word descriptors a numerical value since they aer unique, but the model worked particularly well without needing to do this.

Oranguru is an average Pokémon. It has a total of 490 and has a gender. [https://bulbapedia.bulbagarden.net/wiki/Oranguru_(Pok%C3%A9mon)]

Running the trained models on these new Pokémon returns the following table 05:

	Solgaleo	Kommo-o	Tapu Koko	Oranguru
DecisionTree	True	False	False	False
Perceptron	False	False	False	False
KNN	True	False	True	False
Gauss	True	False	True	False

Table 05: Results of the Generation 1 – 6 trained machine-learning models on 4 new Generation 7 Pokémon. The correct results are highlighted in green and the incorrect results are highlighted in red.

The DecisionTree model incorrectly labelled Tapu Koko as a non-legendary Pokémon and the Perceptron model incorrectly labelled Solgaleo and Tapu Koko as non-legendary Pokémon. Noting these exceptions, the trained models were correct in their labelling of legendary and non-legendary Pokémon.

Concluding Remarks

This report was a lot of fun to write as I explored the data set describing the world of Pokémon (up to and including Generation 6). In doing so, I showcased my skills as a Data Analyst with Python (data visualisation and data manipulation), describing the data thoroughly and concisely through bar charts, pie charts, scatter plots, and tables. There wasn't much scope to showcase my skills as a Data Scientist with Python (Data Analyst + statistical and machine learning techniques) given the nature of the data other than the legendary prediction at the end, but that will be addressed in the following two projects I intend to undertake.

The first of the two follow-up projects I will be undertaking is the beginner Kaggle competition: "Titanic: Machine Learning from Disaster", detailed here:

https://www.kaggle.com/c/titanic

For those unaware, the task presents you with a data set detailing the passengers on board the Titanic and asks you to make a binary prediction on who survived and who did not, similar to my Legendary or Non-Legendary classification above.

The second project I intend to undertake will be an exploration into a game of chance. This will be far more mathematical and include both an analytical calculation using Markov chains and an application of "Hacker Statistics" by calculating the probability in a Monte Carlo simulation by playing the game many 100's of 1,000's of times.

Until then; Goodbye, World!

About the Author

Hello, World! My name is, Matthew, and I am beginning my journey into the world of Data Science. If you wish to reach me, please select from the following options:

E-Mail Address and Personal Website

E-Mail: Matthew.William.Noble@gmail.com
Personal website: http://matthewnoble.info/index.html

CV and Coding Portfolio

CV: http://matthewnoble.info/MyCV.html
GitHub: https://github.com/MatthewWilliamNoble/CodingPortfolio

Social Media

Facebook: https://www.facebook.com/matthew.w.noble
LinkedIn: https://www.linkedin.com/in/matthew-w-noble/
Twitter: https://twitter.com/Matthew_W_Noble
Instagram: https://www.instagram.com/matthew.w.noble/
Kaggle: https://www.kaggle.com/matthewwilliamnoble
HackerEarth: https://www.hackerearth.com/@matthew.william.noble
HackerRank: https://www.hackerrank.com/Matthew_W_Noble

Appendix A

Pokémon type distributions of the combined type 1 and type 2, type 1 alone and type 2 alone for all six Pokémon generations.

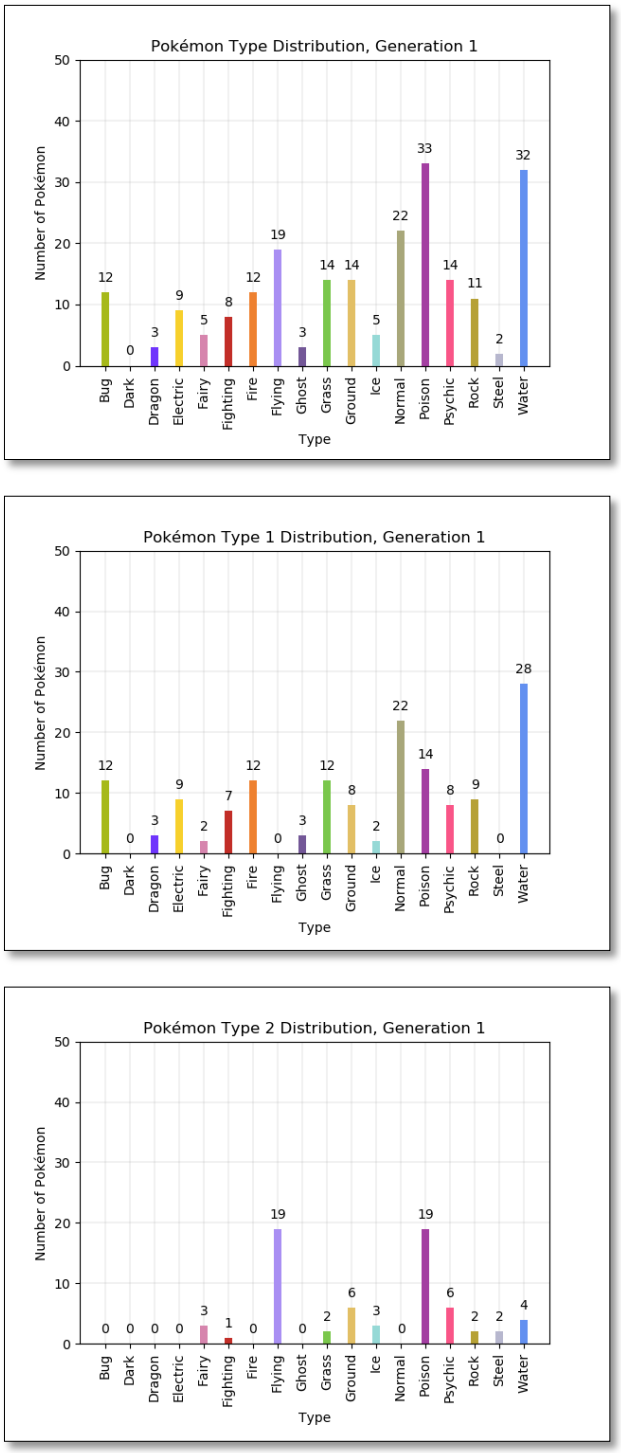


Figure 12: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 1. Plots created using `TypePlot.py`

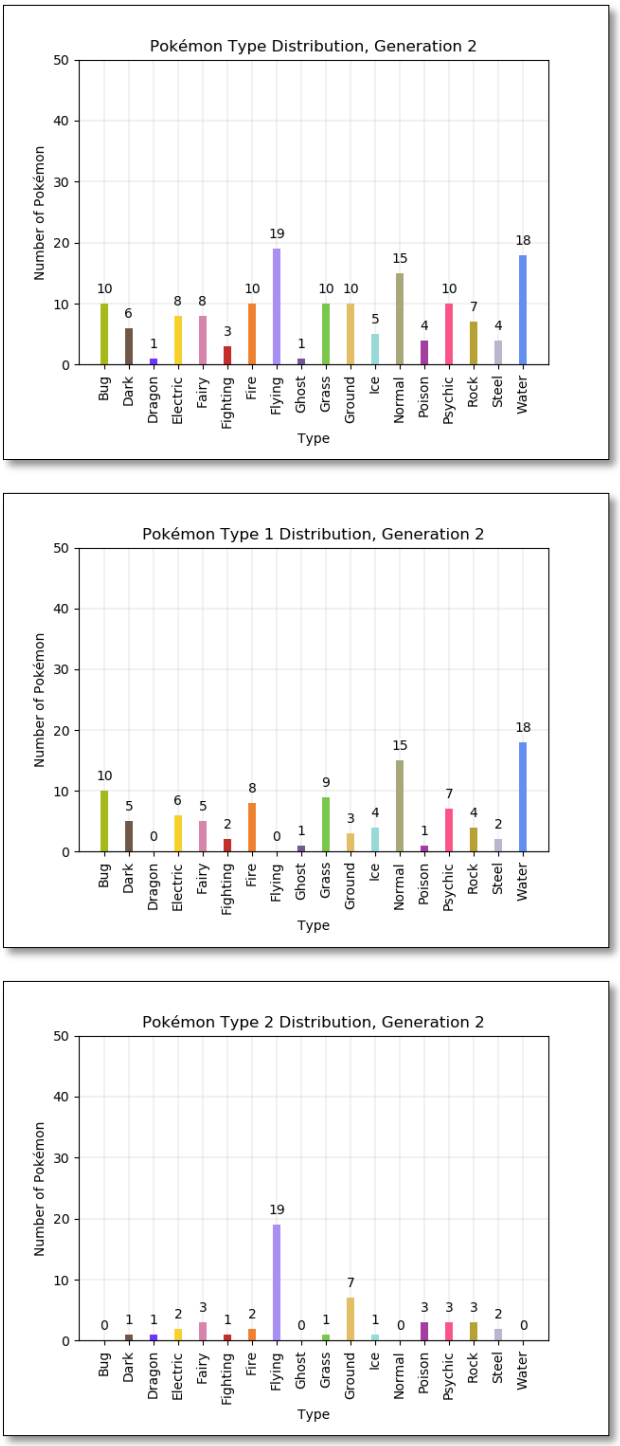


Figure 13: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 2. Plots created using `TypePlot.py`

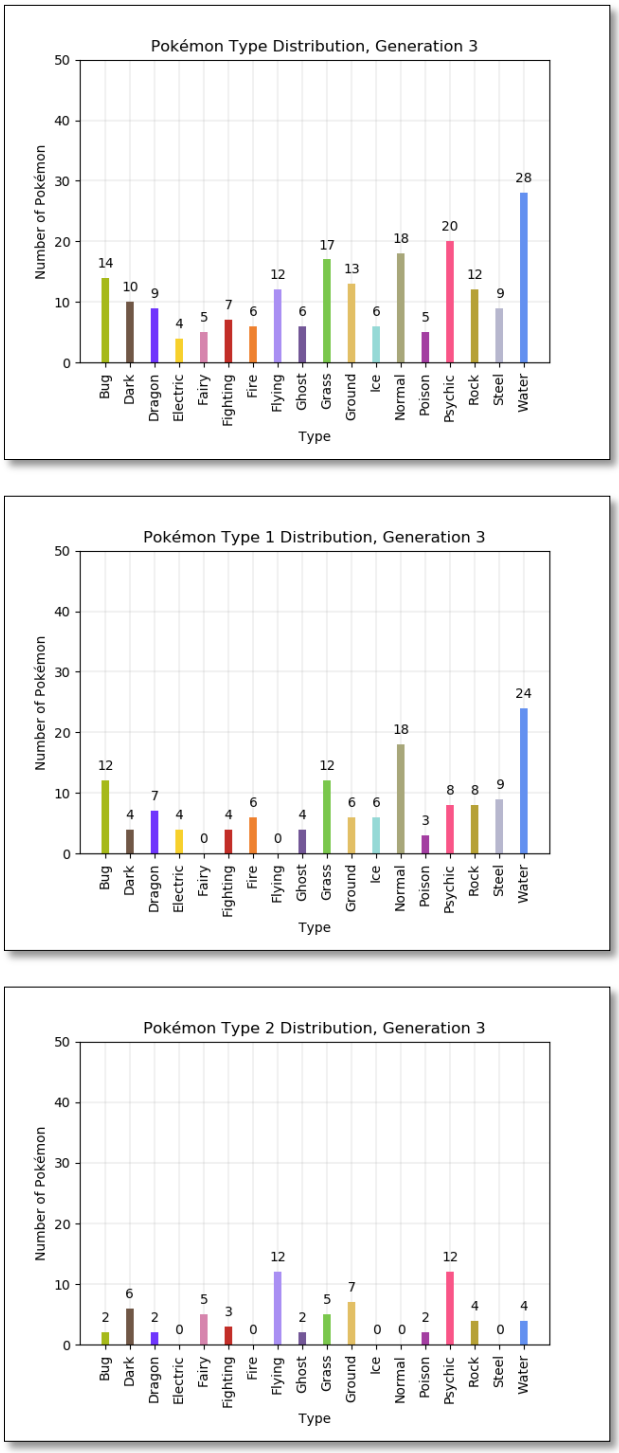


Figure 14: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 3. Plots created using `TypePlot.py`

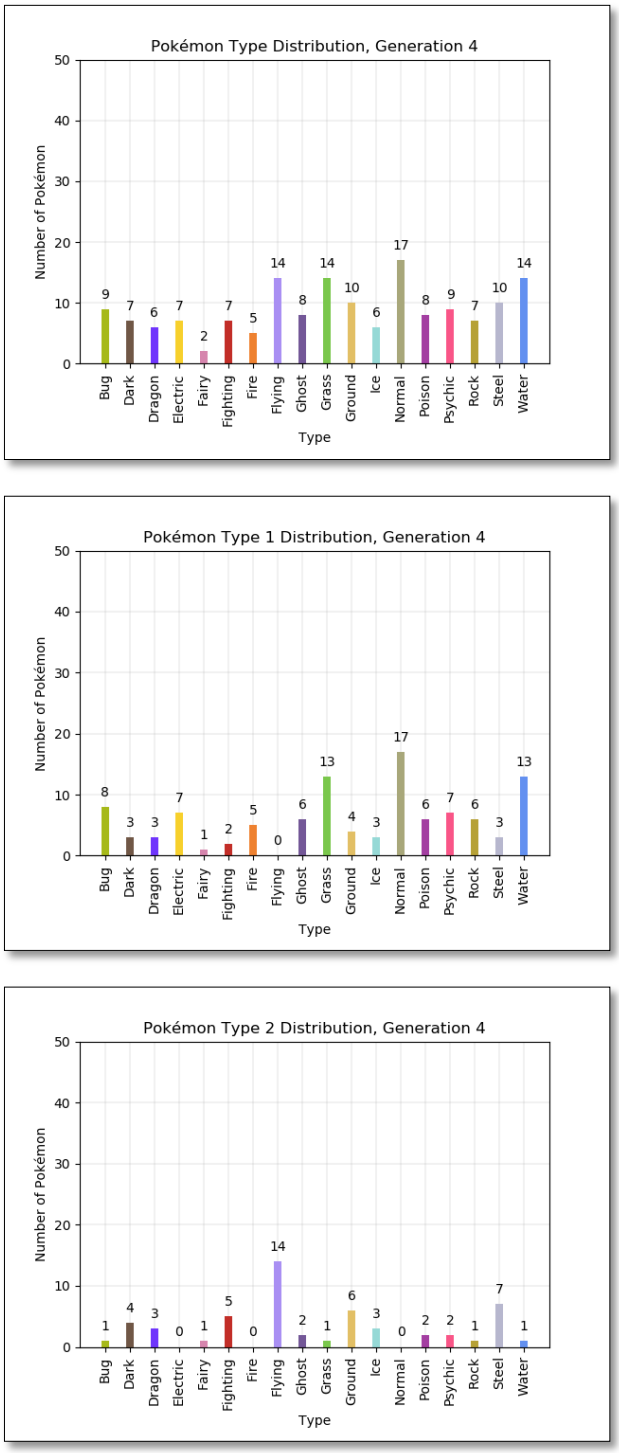


Figure 15: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 4. Plots created using `TypePlot.py`

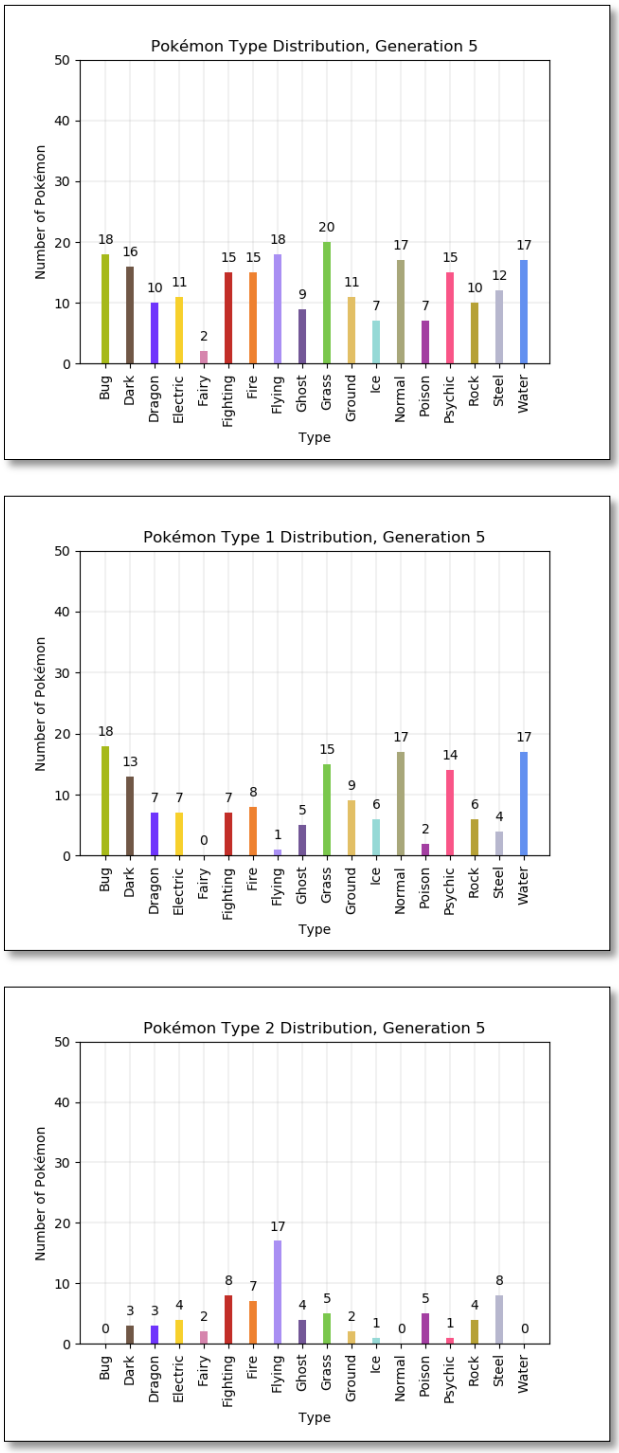


Figure 16: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 5. Plots created using `TypePlot.py`

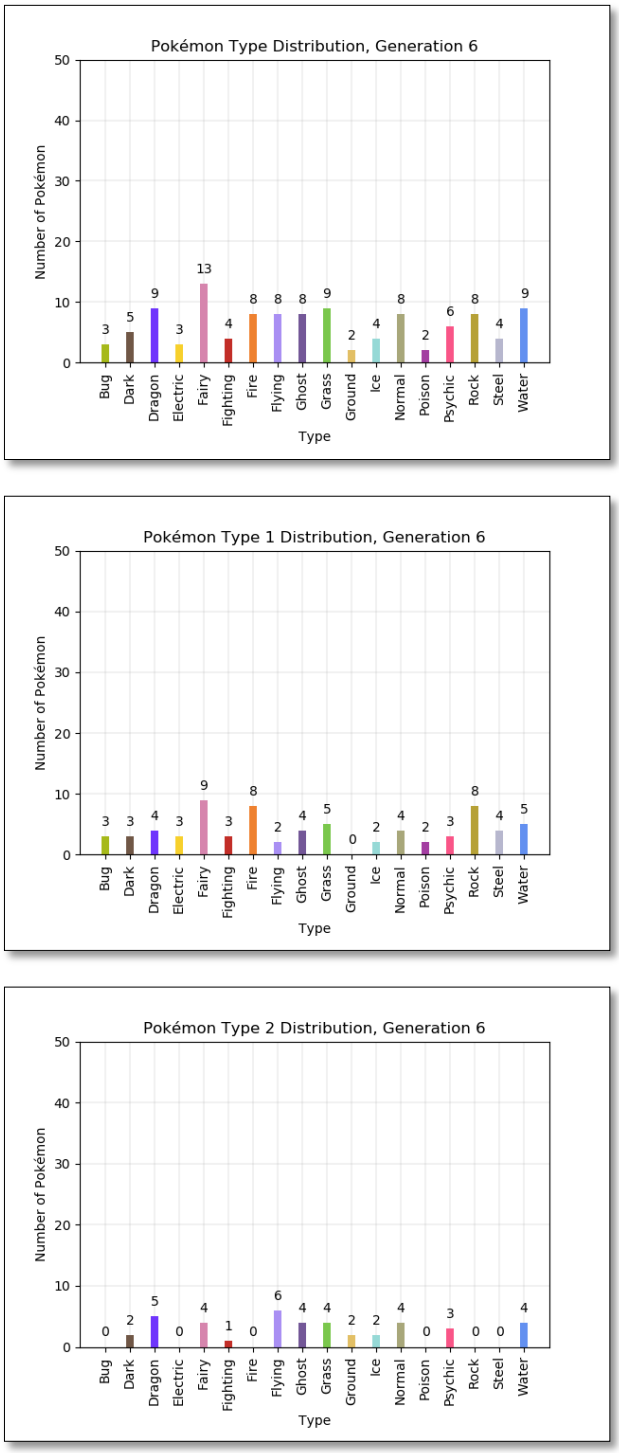


Figure 17: Colour-co-ordinated bar charts showing the combined type, type 1, and type 2 distributions of Pokémon types for generation 6. Plots created using `TypePlot.py`